Incremental learning algorithm for spike pattern classification

Mohemmed, A ; Kasabov, N

Abstract: In a previous work (Mohemmed et al.), the authors proposed a supervised learning algorithm to train a spiking neuron to associate input/output spike patterns. In this paper, the association learning rule is applied in training a single layer of spiking neurons to classify multiclass spike patterns whereby the neurons are trained to recognize an input spike pattern by emitting a predetermined spike train. The training is performed in incremental fashion, i.e. the synaptic weights are adjusted after each presentation of a training pattern. The individual neurons are trained independently from other neurons and on patterns from a single class. A spike train comparison criterion is used to decode the output spike trains into class labels. The results of the simulation experiments on a synthetic dataset of spike patterns show a high efficiency in solving the considered classification task.

DOI: https://doi.org/10.1109/IJCNN.2012.6252533

Originally published at:
Mohemmed, A; Kasabov, N (2012). Incremental learning algorithm for spike pattern classification. In: IEEE World Congress on Computational Intelligence (WCCI 2012), Brisbane, Australia, 10 June 2012 - 15 June 2012. IEEE, 1227-1232.
DOI: https://doi.org/10.1109/IJCNN.2012.6252533

Incremental Learning Algorithm for Spatio-Temporal Spike Pattern Classification

Ammar Mohemmed
Knowledge Engineering and Discovery Research Institute
Auckland University of Technology
120 Mayoral Drive, Auckland, New Zealand
Email: ammar.mohemmed@aut.ac.nz

Nikola Kasabov
Knowledge Engineering and Discovery Research Institute
Auckland University of Technology
Institute for Neuroinformatics
ETH and University of Zurich
Email: nkasabov@aut.ac.nz

***Abstract*—In a previous work (Mohemmed et al. [11]), the authors proposed a supervised learning algorithm to train a spiking neuron to associate input/output spike patterns. In this paper, the association learning rule is applied in training a single layer of spiking neurons to classify multiclass spike patterns whereby the neurons are trained to recognize an input spike pattern by emitting a predetermined spike train. The training is performed in incremental fashion, i.e. the synaptic weights are adjusted after each presentation of a training pattern. The individual neurons are trained independently from other neurons and on patterns from a single class. A spike train comparison criterion is used to decode the output spike trains into class labels. The results of the simulation experiments on a synthetic dataset of spike patterns show a high efficiency in solving the considered classification task.**

I. Introduction

One recognized feature of the neurons in the brain is the use of spikes to exchange information. How the spikes are utilized by the neurons to communicate is an important field in neuroscience research [5], [16]. Understanding the neural code helps in understanding how the brain performs its computation functions and also in emulating it in order to design more efficient artificial intelligent methods.

It has long been assumed that the information is represented in the firing rate [4]. However, over the last two decades a large volume of physiological and behavioral data has emerged supporting a key role for temporal coding in the brain [2], [15], [17]. For example, the temporal coding is known to be effective in information transmission in the visual system. The neurons in the early visual system from the retina to the Lateral Geniculate Nucleus (LGN) to area V1 produce very precise spike sequences in response to a repeated input stimulus that cannot be interpreted within the formulation of the firing rate [1], [15], [14]. Additionally, some pattern recognition tasks such as colors, visual patterns, odours and sound quality are not possible with rate coding [8].

There is no precise definition of temporal coding but in general it relies on the time of the spikes and on the significance of each spike rather than the number of spikes to carry the information. Temporal coding assumes different formats including the time latency to first spike, the order of the spikes, the inter-spike interval and the precise timing of the spikes [18].

Recently a number of learning algorithms based on temporal coding have been proposed [3], [7], [11], [13]. Despite their algorithmic simplicity, they are proved to be efficient in complex spike computation tasks including precise time spike sequence generation and spike pattern classification. These algorithms are based on a network architecture consisting of a single spiking neuron stimulated by many synapses. The weights of the input synapses are tuned in a number of training epochs until the neural spike output matches a desired pattern.

In a previous work [11], [12], the authors proposed SPAN, Spike Pattern Association Neuron, a Leaky and Integrate Firing (LIF) neuron that uses a supervised learning rule for input/output spike pattern association. The main idea of SPAN is converting the input, actual output and desired spike patterns into analogue signals by convolving with a real valued function, such as the alpha function, and then applying the Widrow-Hoff rule [20] to adjust the synaptic weights. The main advantages of this rule are its simplicity and its straightforward implementation. With a single neuron, it is possible to associate output spike patterns to complex input spatio-temporal spike patterns. Although the algorithm is efficient in precise time sequence generation, the main envisioned application of the learning rule is spike pattern classification. Because it is based on temporal coding, the class label of a pattern is identified not only by the neuron firing or not firing but also by the precise time of the output spike pattern.

This paper focuses on extending the application of SPAN to training a single layer of spiking neurons to classify spike patterns. The input is given as spatio-temporal spike patterns grouped into a number of categories. A single layer of SPANs is trained to associate each category to a specific target spike sequence.

The training procedure described in this paper has three main differences over our previous work [11]. Firstly, the training is performed in incremental fashion, i.e., the synaptic weights are adjusted after each presentation of training pattern, rather than batch learning, enabling the algorithm to be used for online learning. Secondly, because multiple neurons are used the decoding mechanism of the output spike sequences into class label is based on the time and space distribution of the output neurons adding redundancy to the class deter-

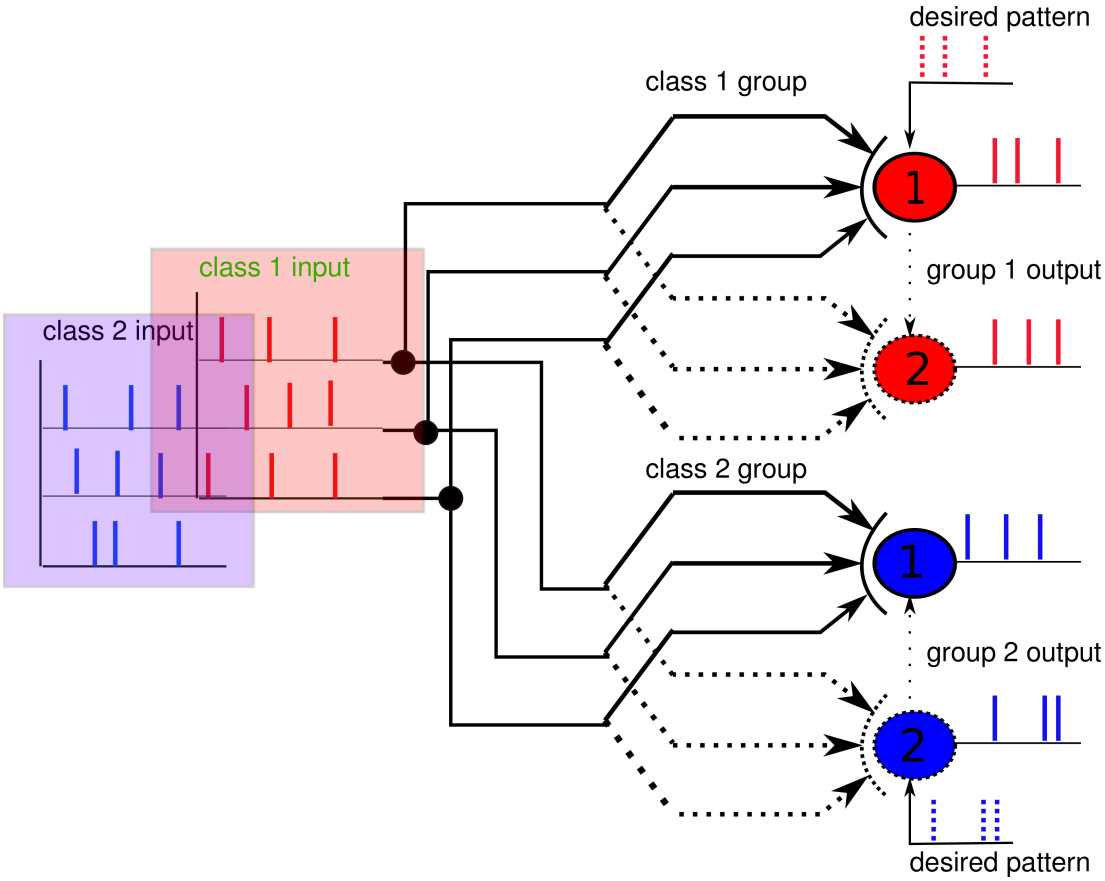

Fig. 1. The architecture of a single layer spiking neural network. The dot lines refer to more neurons can be added to the network to detect different features of the input stimulus.

mination mechanism. Thirdly, the training of the individual neurons is performed independently of other neurons, that is only patterns of a single class are used to adjust the synaptic weights of a neuron.

The paper is structured as follows. The next section briefly describes the SPAN learning rule II. In section III, we discuss the network architecture, the training procedure and the class label mechanism to identify the class label of unseen pattern. Section IV and V present the experimental work and discussion respectively. Section VI concludes the paper.

II. Brief description of SPAN learning rule

For the sake of completion, we briefly describe the learning rule of SPAN and the reader is referred to [11], [12] for more detailed explanation.

The neural model is a Leaky Integrate and Fire (LIF) neuron with membrane potential u described by the following differential equation:

$$\tau_m \frac{du}{dt} = -u(t) + R\ I^{\text{syn}}(t) \tag{1}$$

where $\tau_m = RC$ is the membrane time constant of the neuron, R and C are the resistance and capacitance of the membrane respectively and $I^{\text{syn}}(t)$ is the total current induced by the synaptic inputs.

SPAN is able to associate an input spike pattern to a desired spike pattern by adjusting the weight (w) of the input synapse i in a number of epochs based on Widrow-Hoff learning rule:

$$\Delta w_i^{\text{WH}} = \lambda\ (y_d - y_a)\ x_i \tag{2}$$

where $\lambda \in \mathbb{R}$ is a small real-valued positive learning rate, x_i is the input transferred through synapse i, and y_d and y_a refer to the desired and the actual neural output respectively. To apply 2 on a spiking neuron where the signals are spikes, the input, output and target spike patterns are convolved with the alpha kernel function, other functions also can be used. The final rule for adjusting the synaptic weight is given by (see [12] for a mathematical derivation of the rule):

$$\begin{aligned}\Delta w_i \quad = \quad & \lambda \left(\frac{e}{2}\right)^2 \left[\sum_g \sum_f (|t_i^f - t_d^g| + \tau_s) e^{-\frac{|t_i^f - t_d^g|}{\tau_s}}\right. \\ & \left. - \sum_h \sum_f (|t_i^f - t_a^h| + \tau_s) e^{-\frac{|t_i^f - t_a^h|}{\tau_s}}\right]\end{aligned} \tag{3}$$

where τ_s is the kernel function time constant, e is the exponential constant, t_i, t_a and t_d are the times of the input, actual output and desired spikes respectively,. In [11] the rule was used in a batch learning mode, i.e, in every epoch the all

input patterns are presented to the network sequentially, then the error produced for each training sample is accumulated and applied to compute the synaptic change Δw_i for the next epoch. The next section describes an incremental version of the rule applied on multiple neurons.

III. Incremental learning of single layer SNN

It is known that a single layer feed-forward neural network (Perceptron) consisting of simple threshold or sigmoidal nodes is not capable of solving nonlinear separable problems [10]. However, this limitation can be overcome if the nodes are equipped with more advanced information processing compared to threshold/sigma nodes or increasing the representation dimensionality of the input vectors [19]. To some extent, both of these features are applicable in a single layer network consisting of spiking neurons. The spiking neuron in the simple LIF model is controlled by a system of differential equations and the input consists of high dimensional spike patterns.

In the following subsections we discuss the application of SPANs learning rule to train a single layer feed-forward SNN. The envisioned computation task is that of spatio-temporal spike pattern classification. The inputs are class labeled spike patterns, each pattern consists of a number of spike trains. The spike patterns are assumed to come from specific labeled data (speech, video, etc.) converted into spikes. The conversion mechanism of raw analogue data into spike patterns is an important issue that is left for future investigation. The objective of the task is to train a single layer of SPANs to classify the spike patterns into their categories by firing a target spike train assigned to each class.

A. Training procedure

The architecture of the network is depicted in Fig. 1. The network consists of a number of SPAN groups equal to the number of classes, and each group consists of one or more of SPANs. A single group of neurons is assigned to identify one class. Each neuron is trained to fire a specified target spike sequence when a spike pattern of a certain class is presented at its input synapses. The neurons are connected fully to the input nodes. The question is why more than single SPAN per class exist. The answer lies in increasing the redundancy to the criteria of deciding the class label of the unseen patterns. For example, the neurons of a single group can be trained to emit different target spike trains to better capture the time structure of the input pattern, or to train the neurons on different features of the stimulus.

We note that each neuron of a group class is trained on its class patterns independently from other neurons, i.e. the synaptic weights of a single neuron are affected only by its respective class patterns.

Because SPAN's learning rule is based on the Delta rule, each neuron is trained by incremental learning steps. In a single epoch, the input patterns that belong to a single class are presented sequentially to a single SPAN, where after each pattern presentation the produced spike trains and the target spike trains are used to change the synaptic weights according to Eq. 3.

B. Decoding the spike sequence output

The second issue is how to interpret the spike sequence output of the trained network into a class label to identify an unseen spike pattern. The output of each neuron is a spike train and there are multiple neurons for each class providing a space-time relationship to the decoding criteria. The spatial factor is represented by the neuron group and the time factor is represented by the time accuracy of the output sequence with respect to the desired sequence. A number of approaches can be followed to decide the class label of a pattern. In this paper we adhere to the following procedure.

We measure the dissimilarity between the actual sequence output of a neuron and its assigned target pattern using Eq. 4; once the average error for each neuron group has been calculated, the neuron group that has the minimum error will claim the class label of the pattern. The dissimilarity formula is given by:

$$E = \int_0^\infty |\tilde{y}_d(t) - \tilde{y}_a(t)| \text{ dt} \tag{4}$$

where $\tilde{y}_d$ and $\tilde{y}_a$ are the kernel convolved signals of the desired and actual spike patterns respectively. For example, assuming there are three neuron groups (G1, G2, and G3) corresponding to three classes, each group consists of two SPANs. Assuming that after presenting the network with an input spike pattern, the computed errors, i.e. the difference between the actual output train and the target train, of the three groups are E1 = $\{50., 60.\}$, E2 = $\{40., 80.\}$ and E3 = $\{100., \zeta\}$, where ζ is a large number given as error for the neuron which does not spike. Given these error values, the input pattern is decided to be of class E1, since the average error of this group is the smallest. This criterion gives the best results for our experiments although other approaches such as adopting the class of the SPAN that produces the smallest error among the all neurons, could be followed.

IV. Experimental work

We performed a number of simulation tests to investigate the performance of the learning rule in a classification task. We measure the performance of the network in term of the classification accuracy, i.e., the percentage of spike patterns that are classified correctly in the testing phase. The number of testing patterns is fixed to 100 per class. The simulation is conducted using NEST [6]. Each experiment is repeated for 20 runs, where each run has different synaptic weights and different input patterns initialized randomly. Other parameters of the simulation and the spiking neurons are listed in Table I.

A. The input dataset

The following procedure is used to generate the spike patterns to train and evaluate the network. Firstly, a number of spike patterns equalling the number of classes are generated randomly based on a uniform distribution. Each pattern

TABLE I
TABULAR DESCRIPTION OF THE NEURAL MODEL AND EXPERIMENTAL SETUP.

Model Summery	
Neural model	Leaky integrate-and-fire
Synaptic model	α shaped synaptic currents
Input	Random input
Connectivity	All input neurons are connected fully to the output neurons
Neural Model	
Type	Leaky integrate-and-fire (LIF) neuron
Description	Dynamics of membrane potential $u(t)$: • Spike times: $t^{(f)} : u(t^{(f)}) = \vartheta$ • Sub-threshold dynamics: $\tau_m \frac{du}{dt} = -u(t) + R\ I^{\text{syn}}(t)$ • Reset & refractoriness: $u(t) = u_r \forall f : t \in (t^{(f)}, t^{(f)} + \tau_{\text{ref}})$ • exact integration with temporal resolution dt
Parameters	Membrane time constant $\tau_m = 10$ms Membrane resistance $R = 333.33\text{M}\Omega$ Spike threshold $\vartheta = 20$mV Reset potential $u_r = 0$mV Refractory period $\tau_{\text{ref}} = 3$ms Time resolution $dt = 0.1$ms Simulation time $T = 200$ms
Synaptic Model	
Type	Current synapses with α function shaped post-synaptic currents (PSCs)
Description	Synaptic input current $I^{\text{syn}}(t) = \sum w \sum_f \alpha(t - t^{(f)})$ $\alpha(t) = \begin{cases} e\ \tau_s^{-1}\ t\ e^{-t/\tau_s}, & \text{if } t > 0 \\ 0, & \text{otherwise} \end{cases}$
Parameters	Synaptic weight $w \in \mathbb{R}$, uniformly randomly initialized in $[0, 10]$ The kernel time constant $\tau_s = 8$ms

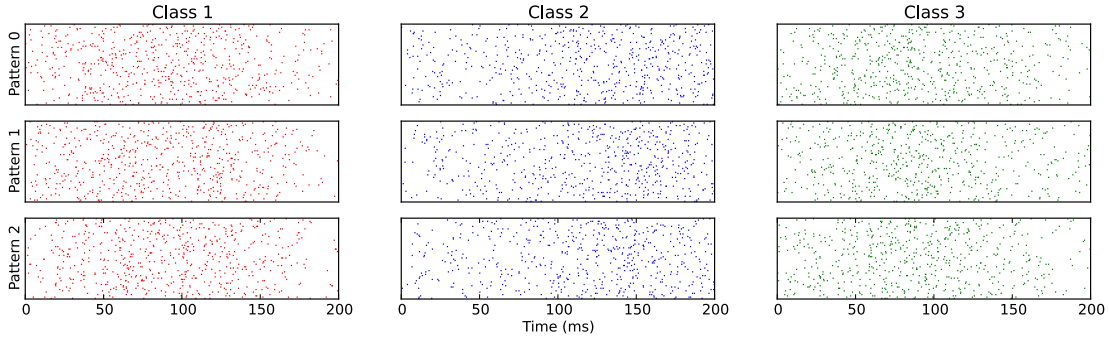

Fig. 2. Example of spike patterns generated for training. The patterns are generated by adding a Gaussian time jitter of zero mean and 30 ms standard deviation to base patterns generated from a uniform distribution, see section IV-A.

consists of 64 spike trains, in which each train has 10 spikes chosen randomly in the period of simulation. From each base pattern, a number of training patterns are created by adding a time jitter to each spike. The jitter is drawn from a Gaussian distribution with a mean value of zero and a specified variance. Following the same procedure, a set of unseen patterns is created for each class to test the generalization of the network. To give a visual impression of the created spike patterns, three classes with three patterns generated by the mentioned procedure are displayed in Fig. 2.

B. Experiments

The following tests are conducted to evaluate the performance of the network.

Test 1: Number of Training Patterns. This test investigates how the number of training patterns affects the classification accuracy. It is known that if the number of patterns is small, over-fitting on the training patterns may occur, but if the number of training samples is large then the network might tend to forget previously seen patterns. The training dataset consists of three classes, hence the network constitutes three neurons, each assigned to identify a single class by producing a target spike train $= \{132., 142., 155., 165.\}$ ms in response to the pattern that belongs to its assigned class. The timing of the target spikes is set arbitrary after a few trial and error attempts, but in general the target spikes are located at the end of the spike pattern period. The number of training patterns is changed from 1 to 50 in 5 patterns steps. The accuracy results of this test on the testing patterns are shown in Fig. 3.

Fig. 3 shows the average accuracy of classification for three

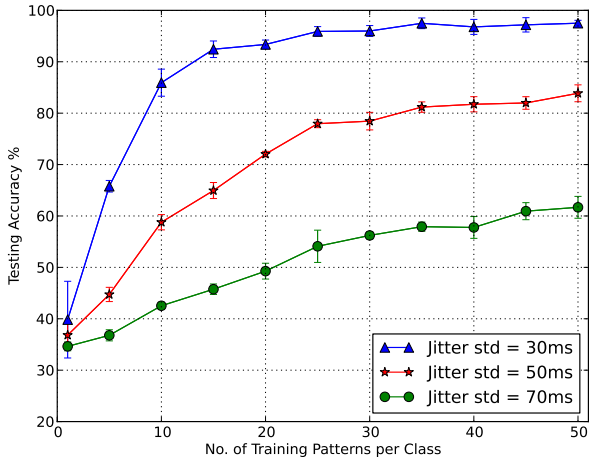

Fig. 3. The classification accuracy vs. the number of training patterns per class for three datasets of different amount of jitters.

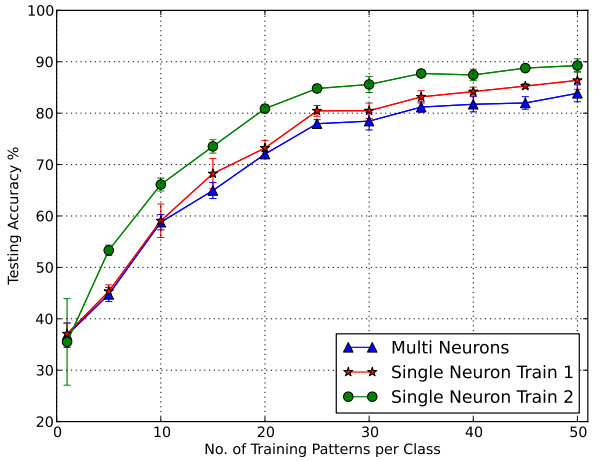

Fig. 4. Accuracy performance vs. number of SPANs per class (Test 2). The (Single Neuron Train 1) and (Single Neuron Train 2) are accuracy plots for networks with single output neuron per class but with different output spike sequence. The (Two Neurons) plot is for a network with two neurons per class.

cases that differ in the amount of jitter added to generate the training patterns: jitters with standard deviation of 30ms, 50ms, and 70ms. Clearly, the network efficiently recognizes the unseen patterns with accuracy of 97% when the jitter is 30%, reducing to 60% when the jitter is 70ms. Because the training is performed in a single presentation of the training patterns, a sufficient number of patterns is required to train the network to achieve acceptable generalization ability. For the given setup and with a jitter of 30ms, 30 patterns per class were required.

Test 2: Number of Neurons per Class. This test is performed to check the impact of increasing the number of output neurons trained to classify a single class. We compare the accuracy when two neurons are assigned to each class with that of a single neuron per class.

In the case of two neurons, the first neuron is trained to produce the same previous spike sequence of Test 1: $= \{132., 142., 155., 165.\}$ ms, and the second one is trained to produce a different sequence: $\{125., 147., 159., 170.\}$ ms. The training dataset consists of 50 patterns per class generated from a jitter of 50ms. The accuracy of this test is shown in Fig. 4.

The figure shows that the two neurons network has better generalization accuracy than a single neuron. Changing the time of the target spikes (Single Neuron Train 1 vs. Single Neuron Train 2 in Fig.5) influenced the performance; the second target train is better aligned to capture the time structure of the input which is reflected in a better accuracy. Combining two neurons each having different output spike patterns improved the accuracy over the performance of single neurons. However, training two neurons is computationally more expensive than a single neuron as the number of synapses to be trained is doubled.

Test 3: Number of Classes. This test examines the effect of increasing the number of classes on the classification performance. Because the training of the neurons is independent, increasing the number of classes could increase the chance of misclassification since no mechanism is used to inhabit the other neurons when one of the output neurons fires. The network uses two SPANs per class and the patterns are generated by adding a jitter of 30.ms. The accuracy of classification for 3 and 6 classes is shown in Fig. 5.

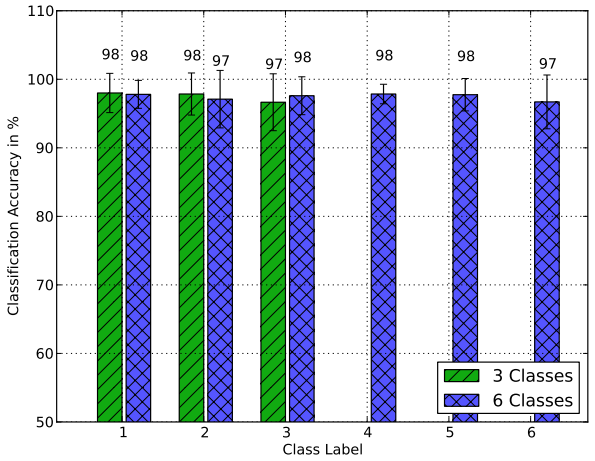

Fig. 5. Accuracy performance vs. number of classes (Test 3).

Increasing the number of classes did not produce a noticeable change in the overall accuracy of classification (97.5% for 3 classes compared with 97.4% for 6 classes). Also, we have tested a 12-class network in which the average accuracy was found to be 95%. However, in a separate test with patterns generated from a jitter of 50ms, we noticed that increasing the number of classes caused a significant decrease in accuracy.

Thus, the network has a certain capacity to perform the classification properly.

V. Discussion

A number of issues arose from applying the SPAN learning rule to training a single layer SNN. First, the training is performed independently on each neuron. During training a single SPAN learns only the patterns of its class. This is possible because during training many synapses will adapt to lock on the temporal structure of the corresponding class patterns and will automatically reject, by firing differently, other patterns. However, this property might highly depend on the intraclass separation of the patterns.

The second issue is deciding properly the spike times of the target trains in order to more accurately classify the given dataset. Here, the times of the target spikes were selected arbitrarily after few trials and were preferentially chosen to be near the end of the pattern time period.

The third issue is decoding the class label from the output spike sequences. As pointed out in the Introduction, biological neurons produce very similar spike sequences in response to the same input stimulus, but it is not clear how these sequences are associated with different categories of the input stimulus [9]. The proposed criterion, which might be biologically implausible, is based on measuring the strength of association between the actual output and the target trains. The multiple neurons assigned to each class can have not only different target trains but different parameters, to give more flexibility in tuning the neurons to recognize specific classes and reduce misclassification. One advantage of this could be training the neurons to detect different features of the input stimulus. However, further study is needed to analyze how and to what extent adding more neurons can assist in improving the performance.

It is noted that the obtained accuracy is higher than that obtained in our earlier paper [11]. This is attributed mainly to the classification criterion used in the present work which is more relaxed and aimed at achieving higher classification accuracy, rather than producing very precise spike sequences. In addition, the network architecture consists of several SPANs rather than a single SPAN to perform the classification as was done in [11]. Using a spike sequence instead of single spikes to encode the class label further improves the performance.

VI. Conclusion

This paper investigated the plausibility of using the precise timing of spikes to perform spike pattern classification. The network consists of a single layer of spiking neurons trained by a supervised learning algorithm [11]. Each neuron is trained incrementally where the synaptic weights are adjusted after each pattern presentation and independently in a single class of patterns. The pattern class of the pattern is determined by the output neurons producing a sequence of spikes rather than firing or not firing. We have demonstrated a potential advantage in training multiple neurons per class with different characteristics such as their output sequence.

Future work will consider applying the algorithm to real-world application in particular to the field of computer vision.

Acknowledgment

The work on this paper has been supported by the Knowledge Engineering and Discovery Research Institute (KEDRI, www.kedri.info). The author, NK, has also been supported by a Marie Curie International Incoming Fellowship Grant with the 7th European Framework Programme under the project "EvoSpike" (http://ncs.ethz.ch/projects/evospike), hosted by the Neuromorphic Cognitive Systems Group of the Institute for Neuroinformatics of the ETH and the University of Zurich.

References

[1] M. J. Berry, D. K. Warland, and M. Meister. The structure and precision of retinal spiketrains. *PNAS*, 94(10):5411–5416, May 1997.

[2] S. M. Bohte. The evidence for neural information processing with precise spike-times: A survey. *NATURAL COMPUTING*, 3:2004, 2004.

[3] R. V. Florian. The chronotron: a neuron that learns to fire temporally-precise spike patterns. http://precedings.nature.com/documents/5190/version/1, Nov. 2010.

[4] W. Gerstner and W. M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, Cambridge, MA, 2002.

[5] W. Gerstner, A. K. Kreiter, H. Markram, and A. V. M. Herz. Neural codes: firing rates and beyond. *Proc. Natl. Acad. Sci. USA*, 94(24):12740–12741, 1997. article.

[6] M.-O. Gewaltig and M. Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

[7] R. Gutig and H. Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nat Neurosci*, 9(3):420–428, Mar. 2006.

[8] J. Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376:33–36, 1995.

[9] G. E. Hugo and S. Ines. Time and category information in pattern-based codes. *Frontiers in Computational Neuroscience*, 4(0), 2010.

[10] M. Minsky and S. Papert. *Perceptrons*. Cambridge, MA: MIT Press, 1969.

[11] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov. Method for training a spiking neuron to associate input-output spike trains. In *EANN/AIAI (1)'11*, pages 219–228, 2011.

[12] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov. Span: Spike pattern association neuron for learning spatio-temporal sequences. *International Journal of Neural Systems*, 2012 (To Apper).

[13] F. Ponulak and A. Kasiński. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural Computation*, 22(2):467–510, Feb. 2010. PMID: 19842989.

[14] P. Reinagel and R. C. Reid. Temporal coding of visual information in the thalamus. *Journal of Neuroscience*, 20(14):5392–5400, 2000.

[15] P. Reinagel and R. C. Reid. Precise firing events are conserved across neurons. *Journal of Neuroscience*, 22(16):6837–6841, 2002.

[16] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek. *Spikes: exploring the neural code*. MIT Press, Cambridge, MA, USA, 1999.

[17] A. Rokem, S. Watzl, T. Gollisch, M. Stemmler, A. V. Herz, and I. Samengo. Spike-timing precision underlies the coding efficiency of auditory receptor neurons. *J Neurophysiol*, 2005. automatic medline import.

[18] F. Theunissen and J. P. Miller. Temporal encoding in nervous systems: a rigorous definition. *Journal of Computational Neuroscience*, 2(2):149–162, 1995.

[19] T. Trappenberg. *Fundamentals of Computational Neuroscience*. Oxford University Press, USA, June 2002.

[20] B. Widrow and M. Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415 –1442, sep 1990.